From:	James Ramsey
To:	Microsoft ATR
Date:	12/27/01 9:13pm
Subject:	Apparent misperceptions about security

One of the sections from the Final Judgment, III.J.1, has already been noted controversial because it is a possible loophole. However, aside from that problem, it also appears to rest on a false understanding of how to make secure software. From III.J.1:

"No provision of this Final Judgment shall ... [r]equire Microsoft to document, disclose or license to third parties: (a) portions of APIs or Documentation or portions or layers of Communications Protocols the disclosure of which would compromise the security of a particular installation or group of installations of anti-piracy, anti-virus, software licensing, digital rights management, encryption or authentication systems, including without limitation, keys, authorization tokens or enforcement criteria; or (b) any API, interface or other information related to any Microsoft product if lawfully directed not to do so by a governmental agency of competent jurisdiction."

The above appears to be built on the idea of "security by obscurity," where security is dependent upon hiding the implementation used to secure something. In the physical world, this would be analogous to making the mechanisms of a lock or safe a trade secret. In the computer realm, this would mean keeping secret the mathematics or algorithms, source code, protocols, etc. of cryptographic software. While this appears to make sense on its face, it has long been discredited by those who deal with computer security, such as Bruce Schneier, author of "Applied Cryptography" and "Secrets and Lies," a book about dealing with real-world security problems. (His business's website, by the way, is http://www.counterpane.com.) In particular, the core problem with "security by obscurity" is that it is fragile, that is, the security implementation is not necessarily obscure to the ones who may attempt to break it. Industrial spies or hackers/crackers have the tools and expertise to discover the source code or algorithms of a piece of security software. Even those who are not "black hats" may break proprietary, secret algorithms with relative ease. (See http://www.counterpane.com/crypto-gram-9902.html#snakeoil)

Much of strong cryptographic and security software, rather than relying on the secrecy of the algorithm or implementation, relies on public algorithms and often public implementations. What is kept secret is a long number, a key, used in combination with the algorithm, and knowledge of the algorithm is useless without the key. Examples of public cryptographic algorithms are the government standards DES (recently "retired") and AES (DES's replacement), and RSA, the algorithm behind SSL, the protocol used for secure Internet transactions. Examples of secure software with public implementations are OpenBSD, OpenSSH, OpenSSL, and PGP.

The point of this discussion of "security by obscurity" is that Microsoft (MS) should have no need to hide the protocols and APIs used for security. Unless their software has a fragile security implementation, disclosing the protocols and APIs should do no damage or compromise security.

The only possible exception to the above points is digital rights management (DRM), which is inherently fragile. (See
http://www.counterpane.com/crypto-gram-0105.html#3)
However, DRM is more designed to deter would-be casual copyright infringers, who lack technical knowledge, rather than mass-scale pirating operations of the kind one sees in Asia. The documentation of DRM APIs and protocols would be of little use to those whom DRM is designed to thwart.

In general, there is no good technical reason to allow Microsoft to have any private APIs.

=====
-- I am a fool for Christ. Mostly I am a fool. --